ONE-SHOT LEARNING FOR SEMANTIC IMAGE SEGMENTATION

A Dissertation
Presented to
The Academic Faculty

By

Zhen Liu

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in Computer Science with the Research Option in the
College of Computing

Georgia Institute of Technology

April 2017

ONE-SHOT LEARNING FOR SEMANTIC IMAGE SEGMENTATION

Approved by:

Dr. Byron Boots, Advisor
College of Computing
Georgia Institute of Technology

Dr. James Hays
College of Computing
Georgia Institute of Technology

Date Approved: May 1, 2017

ACKNOWLEDGEMENTS

I would like to thank my advisor for this thesis, Dr. Byron Boots, for providing me tremendous support and guidance throughout my undergraduate research career. And I want to pay special thanks to the leaders of this project, PhD students Amirreza Shaban and Shray Bansal, without whom this thesis would not be possible. Finally, I would like to thank Dr. Irfan Essa for his precious advices on our project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Image segmentation has been one of the central topics in computer vision. Recent progress in deep learning methods and large-scale datasets provides opportunities for learning high level semantic representation of images. The recent success in one-shot video segmentation, in which the algorithm learns how to track or segment out some object in the video only given the first frame. This thesis aims to design an one-shot learning algorithm for image segmentation task, in which the algorithm learns given one or several segmentation samples.

CHAPTER 1
INTRODUCTION

1.1 Overview

The recent progress in machine learning algorithms, especially deep learning, has introduced new possiblities for solving copmuter vision tasks. With large datasets such as ImageNet [1], various learning models are now capable of tasks such as image classification [2, 3, 4, 5], image segmentation [6], object detection [7, 8, 9, 10], and etc. However, many of their methods rely on large human-labeled datasets, while humans can easily learn new concepts based on one or very few new samples. To solve this issue, the so-called one-shot learning algorithms seek to generalize well from previously learned knowledge given only one sample from the new concept. A closely related field, few-shot learning, generalizes using few samples instead of one.

It has been shown that prior knowledge learned from previously observed information, such as the appreance of natural images and geometry can greatly increase the performance of recognizing new objects [11, 12, 13]. And recent success in one-shot video segmentation has shown possibility that models can generalize well based on limited data. However, little work so far has been done on one-shot learning for image segmentation, a problem harder than one-shot video segmentation since it only requires the objects to be in the same semantic class. My research goal in this thesis is to develop an efficient one-shot learning model for image segmentation tasks.

CHAPTER 2

BACKGROUND

2.1 Fundementals of Deep Convolutional Neural Nets

Deep convolutional neural nets, or CNN, are widely used in computer vision tasks because they are able to extract deep features from images. To introduce the concept of CNN and deep learning, we shall first start with the basic concepts of machine learning.

2.1.1 Machine Learning

Machine learning algorithms learn some concept C from data D related to this concept C given some performance measure. Generally, machine learning can be divided into three categories: supervised, unsupervised and reinforcement learning. The concepts of reinforcement learning and unsupervised learning are not the focus of this thesis and therefore skipped.

In supervised learning, the goal is to approximate some function $f : X \to Y$ given some dataset $\{x_i, y_i\}_i$ where x_i is the data vector and y_i is the corresponding label. When y_i is the index of categories in the data, the task is called classification and is otherwise called regression. In computer vision, classification is more pervasive than regression and we shall give some examples of classification algorithms here.

k-Nearest-Neighbor

k-Nearest-Neighbor, or k-NN, is probably the simplest classification algorithm. In k-NN, the query input x', given some training dataset $\{x_i, y_i\}_{i=1}^N$, is classified as y' if most of the top-k nearest neighbors of x' are of category y'.

This algorithm is actually nonparametric and highly non-linear, which in theory can learn any function given enough data. However, when the data vector x is in high di-

mension, we encounter *Curse of Dimensionality*, that the number of data points we need explode exponentially. Consequently, in high dimensional space, k-NN can easily overfit and fail to generalize well on unseen data points.

Logistic Regression

Despite the name, logistic regression is actually an algorithm for binary classification. Logistic regression solves the following optimization problem:

$$L = \min_{\theta} -\frac{1}{N} \sum_{i=1}^{N} [y_i \log y_i + (1 - y_i) \log(1 - y_i)] \tag{2.1}$$

Here, $y_i = f_\theta(x_i)$ is a binary label, i.e. $y_i \in \{0, 1\}$ where x_i is the corresponding data vector and θ is some set of parameter that parametrize the function f. The quantity $y_i \log y_i + (1 - y_i) \log(1 - y_i)$ is also known as cross entropy and thus the loss function L is also called cross entropy loss.

Logistic regression is among a category of models call *generalized linear models* (GLM), which assumes some level of linearity in the model.

Perceptron

Perceptron is the predecessor of neural networks. Geometrically, perceptron aims to find a line that seperates the data into different regions, each of which shall represents one category. The mathematical model is:

$$f(x) = sign(Wx + b) \tag{2.2}$$

where $x \in \mathbb{R}^n, W \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^n$, $sign(\cdot)$ is the sign function which returns 1 if input is non-negative and 0 otherwise. W is called weights and b is called bias.

Perceptrons are known to be capable of learning linear functions. If the training dataset is linear seperable, the model is geranteed to converge using simple optimization algorithms like stochastic gradient descent.

Multi-layer Perceptron

Perceptron is merely a linear model and not capable of learning complex or non-linear models. An extension to perceptron is multilayer perceptron, another simple model of neural networks.

Perceptron can be viewed as a single "neuron", or some basic computational element. In multi-layer perceptron, we have "neurons" stacked in layers. In particular, for layer i, we define

$$f^{(i)}(x) = \sigma(W^{(i)}x^{(i)} + b^{(i)}) \tag{2.3}$$

where $f^i : \mathbb{R}^m \to \mathbb{R}^n$, $x^i \in \mathbb{R}^m$. σ is called *activation function* and usually non-linear to learn non-linear models. The final output y is thus a composition of $\{f^i\}$, or

$$y = f^{(k)}(f^{(k-1)}(...f^{(1)}(x))) \tag{2.4}$$

If $k > 1$, the layers of computation before the final one are called *hidden layers*.

The functions $\{f^{(i)}(x)\}$ are normally dense, such that $\{W^{(k)}\}$ are dense matrices. In other words, each neuron is connected to all neurons in the previous layer. Layers in which each neuron is connected to all neurons in the previous layer are called *fully connected layers*.

The calculation is non-cyclic, since no input of some neuron is explicitly or implicitly dependent on its output. We call this kind of neural network *feedforward network*.

The optimization of multi-layer perceptron, and many other neural networks, is done by *error backpropogation*, or simply *backprop*. In each iteration of backprop, we compute the predicted label of some batch (can be randomly sampled) of data and compute the average loss E. The partial derivative of E with respect to each parameter in the neural network is computed using the chain rule:

$$\frac{\partial E}{\partial W_{ij}^{(k)}} = \frac{\partial E}{\partial o^{(k)}} \frac{\partial o^{(k)}}{\partial W_{ij}^{(k)}} \tag{2.5}$$

and

$$\frac{\partial E}{\partial b_i^{(k)}} = \frac{\partial E}{\partial o^{(k)}} \frac{\partial o^{(k)}}{\partial b_i^{(k)}} \tag{2.6}$$

The update of these parameters are done in gradient descent fashion:

$$\Delta W_{ij}^{(k)} = -\alpha \frac{\partial E}{\partial W_{ij}^{(k)}} \tag{2.7}$$

and

$$\Delta b_i^{(k)} = -\alpha \frac{\partial E}{\partial b_i^{(k)}} \tag{2.8}$$

where α is called the learning rate. Normally the learning rate is first set to large values and gradually decreased to smaller values. Some other optimization techniques such as momentum may be used.

Multi-layer perceptron is powerful in the sense that it can approximate many functions. In face, universal approximation theorem shows that any layered feedforward neural network with one hidden layer can approximate arbitrary function given infinite number of neurons in the hidden layer. However, multi-layer perceptron can contain thousands of parameters and can be prone to overfitting.

2.1.2 Convolutional Neural Networks

Convolutional neural networks (CNN) are neural nets that use convolutional layers as their skeleton, and are classified as one type of deep learning models. Typically, the inputs of CNN are RGB images and thus 3-dimensional.

Convolutional Layer. Convolutional layer, or in this thesis we abbreviate it as conv layer, perform a convolution operation on input x with some kernel K. Let's assume a 3-dimensional input of which the first one is the dimension of feature vector (in the input layer, the dimension is the dimension of each pixel, normally 3 because of RGB channels)

of each pixel and the rest two are spatial dimensions m and n. Different from fully connected layers where each neuron sees all outputs in the previous layer, each neuron in a conv layer can only sees a local area in the output of the previous image. This local area is known as the *receptive field*. In addition, it is worth noticing that the every receptive field is multiplied by the same kernel, which approach is known as *weight sharing*. The local receptive field and weight sharing utilize the fact that images are normally translational invariant and to some extent scale invariant, thus save the number of parameters and prevent overfitting.

Noticably, the use of convolution does not require the shape of input to be some fixed number, while the fully connected layer is (since the shape of the weights and biases are fixed).

Pooling Layer. The idea of pooling is to reduce the number of outputs of the convolutional layer both to save memory and to prevent overfitting. The pooling layer takes the input image or feature map, applies "pooling" operation on each rectangle of some size (non-overlapping, normally square region) in the input and output only one value from each of these rectangles. In Max-Pooling, we take the maximum value out of every rectangular region and in Average-Pooling, we take the average value.

Non-linear operations. Non-linear operations are normally required for CNN to learn non-linear features. The typical choices of the non-linear operations are, but not limited to,

- **Sigmoid**
- **Tanh**
- **Rectified Linear Units (ReLU)**
- **Leaky ReLU**

Among these non-linear operations, ReLU is the most popular one in the recent years because it is linear and thus differentiable except for at origin. This feature enables faster

computation and avoids the vanishing gradient problem which appears in saturated functions like sigmoid.

To learn models for image classification tasks, the output of convolutional layers, together with some ReLU and pooling layers, is followed by some classifiers like multi-layer perceptron (which consists of fully connected layers and a final loss layer).

Empirical evidence shows that given large datasets such as ImageNet [1] CNN can learn deep and distributed representations of images [2, 3, 14], despite its large number of parameters. Some theories are being developed to explain the success of CNN and other deep learning methods [15]. Since theory is not our najor concern here, we assume that CNN is able to learn deep features or embeddings from images.

2.2 Progress in Image and Video Segmentation

2.2.1 Fully Convolutional Network

While CNN along with fully connected layers works well on image classification [2], the spatial dimensions are thrown away in fully connected layers which is not desirable for image segmentation tasks. Long *et al.* proposed *fully convolutional neural nets* (FCN), in which they replace fully connected layer by *fully convolutional layers*. Since all the layers now have spatial dimensions, the problem of image segmentation becomes a pixel-wise classification problem in which every pixel is classified with some local information from feature maps of both coarse and fine levels in the FCN.

2.2.2 One-shot Video Segmentation

Recently, Caelles *et al.* proposed one-shot video object segmentation (OSVOS) [16] for video segmentation task. Their approach consist of three stages: foreground branch, contour branch and boundary snapping. In forground branch, they finetuned the network (pretrained on training set of DAVIS dataset) on only the first frame of some video sequence. This finetuned network can obtain over 77% average intersection over union (meanIoU).

With the contour information from the contour branch, they used boundary snapping to improve the results of the foregound branch and obtained nearly 80% meanIoU on DAVIS test dataset.

A following work, done by Khoreva *et al.* [17], further shows the capablity of deep CNN models to learn how to segment objects in video. In their approach, they trained a un-pretrained FCN, with some data augmentation, to "overfit" the network on only the first frame of the video. Surprisingly, even though the FCN is supposed to be highly prone to overfitting, it can still generalize well on later frames of video sequences. This validates the efficiency of learning deep representations for image and video segmentation.

CHAPTER 3

ONE-SHOT IMAGE SEGMENTATION

3.1 Problem Definition

Let's define the support set $S = \{(I_s^i, Y_s^i(l))\}_{i=1}^k$, a set of image-mask pairs and l as the index of the semantic class. $Y_s^i \in L_{test}^{H \times W}$ is the multi-class segmentation annotation (same spatial dimension as of I_s^i) for image I_s^i. $L_{test} = \{0, 1, 2, ..., p\}$ is the set of indices of semantic classes used in testing where we always assume 0 as the index for background. $Y_s^i(l)$ is the binary mask of I_s^i for semantic class l such that every pixel (x, y) in $Y_s^i(l)$ is equal to 1 if (x, y) in Y_s^i is l, or 0 otherwise. Given this support set S, the goal is to correctly generate a binary mask for semantic class l for a query image I_q. An illustration of the problem for $k = 1$ is given as Figure 3.1.

In the training phase, the model receives $k+1$ image-mask pairs $\{(I_s^i, Y_s^i(l))\}_{i=1}^{k+1}$ where $Y_s^i \in L_{train}^{H \times W}$ is the semantic mask for the training image I_s^i, the first k paris of them ($\{(I_s^i, Y_s^i(l))\}_{i=1}^k$) are used as the support sets and I_s^{k+1} is the query image. The loss between the predicted binary mask $\hat{Y}_s^{k+1}$ and Y_s^{k+1} is computed during training. We set $L_{train} \cap L_{test} = \{0\}$, which contains only the background. All our foreground semantic classes in training phase are different from those in test phase.

One of the major differences between our problem and k-shot image classification [12, 13] is that, objects of semantic classes in L_{test} can appear in training images, as semantic classes in L_{test} can be background in training support sets.

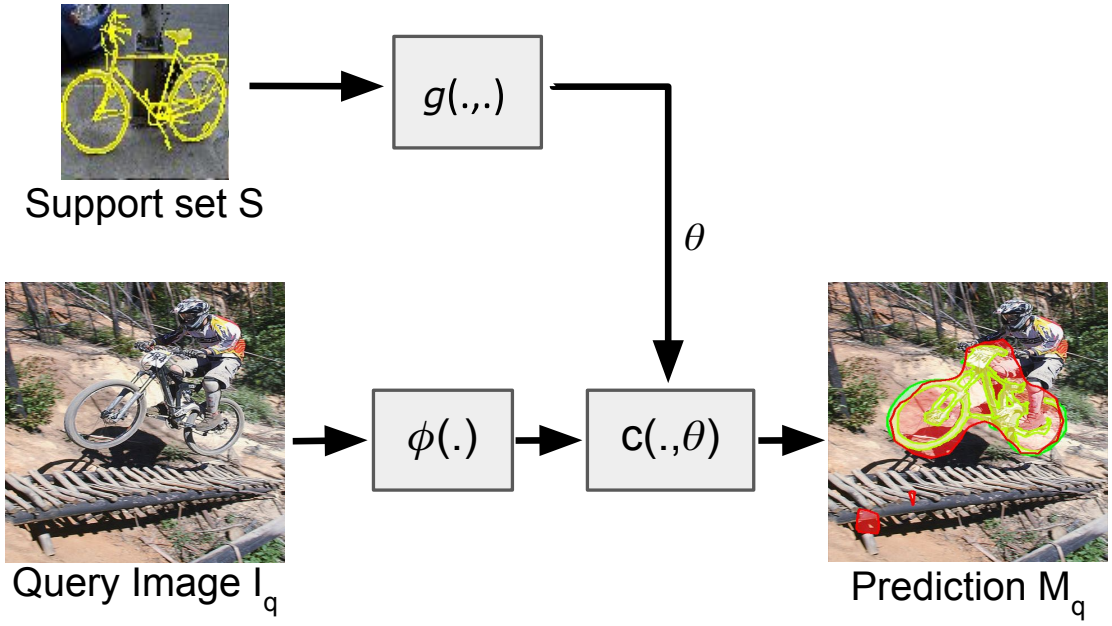

Figure 3.1: Problem Definition. S is the support set, a small batch of annotated images from a new semantic class. In the proposed model, the neural net $g(\cdot,\cdot)$ accepts S as the input and generate the classifier parameter θ. The neural net $\phi(\cdot)$ accepts query input I_q and generates image features $\phi(I_q)$. The classifier c generate predictied binary mask $c(\phi(I_q),\theta)$ for the semantic class in the support set.

3.2 Proposed Method

Hereby we designed a two-branched network architecture. The first branch, given network parameters, generates the embedding of the query image:

$$F_q = \phi(I_q) \tag{3.1}$$

where F_q shall be a 3-D tensor and $F_q(m,n)$ is the feature vector of pixel (m,n).

The second branch, given the support set S, generates the set of parameters W, b of some classifier:

$$W, b = g(I_q) \tag{3.2}$$

The predicted binary mask is generated by pixel-wise logistic regression:

$$\hat{M}(m,n) = \sigma(W^T F_q(m,n) + b)$$

where σ is the sigmoid function, $\hat{M}(m,n)$ is the binary mask predicted at pixel (m,n). In other words, the model predicts the binary mask of each pixel based on the feature vector at the pixel.

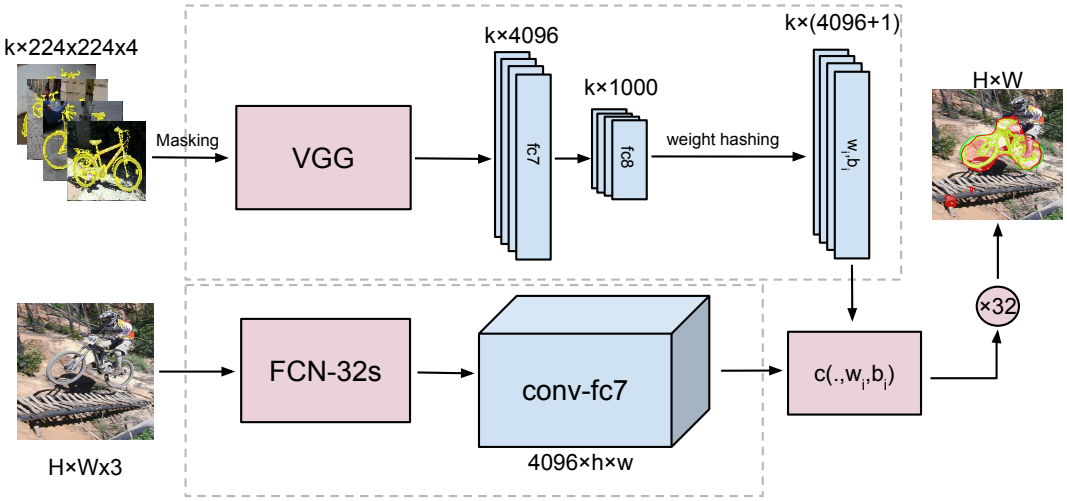

Figure 3.2: Model Architecture. The first branch (top dashed box) receives a batch of k image-label pairs and produces a set of set of k parameters for the ensemble of k logistic regression layers $c(\cdot, w_i, b_i)$. The second branch (bottom dashed box) is a Fully Convolutional Network that receives the query image as input and outputs strided features of convfc7. These pixel-level features are classified by an ensemble of k convolutional layers $c(\cdot, w_i, b_i)$ and generate k different mask predictions. the final binary mask is computed by a channel-wise OR pooling layer.

3.2.1 Generating Classifier Parameters from Support Set

The naive approach for generating classifier parameters from support sets is to use fully connected layers to regress classifier parameters on the image embedding of images in support sets, where the image embedding is generated by some convolutional neural net architecture such as VGG-16 [3]. However, given high dimensional image embedding, the number of parameters in the fully connected layer will be large and the network is therefore prone to overfitting.

Instead of the naive approach, we proposed the following method to modify the VGG-16 architecture from [3] to model the function $g_\eta(\cdot, \cdot)$.

Masking. We masked out the image with the correponding semantic label of the support sets, so that the pixels in images of the support sets are either in the semantic class or in the background. This design is motivated by the following reasons:

(1) This masking solves the overlapping issue, that semantic classes in L_{test} can appear in training support sets as backgrounds.

(2) The network is biased towards the largest object, regardless of the mask.

Weight Hashing. To avoid the problem of overfitting in using fully connected layers while preserving expression power of the model, We used weight hashing [18] to map the feature vector output from the fc-8 layer in VGG-Net to the vector of $\{w, b\}$ concatenated. Weight hashing in [19] randomly copies coefficients, with random sign flipping, of the input vector to obtain the output vector. Specifically, the output vector z, given input vector x, is determined by,

$$z(i) = s(i)x(p(i)) \tag{3.3}$$

where $s(i)$ is subject to a Bernoulli distribution with probability 0.5, and p randomly maps i to $\{1, 2, ..., dim(x)\}$ with equal probability.

The weight hashing in [19] in implicit for memory concerns, we proposed another way to implement weight hashing. Specifically, we used a fully connected layer to explicitly obtain the output vector z and hashing value s, to compute the forward and backward propagation. The weights of the fully connected layer is set as

$$W(i, j) = \begin{cases} s(i) & , p(i) = j \\ 0 & , Otherwise \end{cases} \tag{3.4}$$

3.2.2 Feature Map Extraction of Query Image

Different than the embedding (fc-8 in VGG-Net) used in generating classifier parameters, we used FCN-32 [6] and extract the second last layer as the embedding of query image. The output feature map of this query image, jointly with the classifier parameters $\{W, b\}$, is used to compute the pixel-wise logistic classification results. For performance comparison, we also performed experiments using slower, but with higher resolution, dilated-FCN [20] with $stride = 8$.

3.2.3 Training Procedure

One-shot

In each iteration, with support set S, query image I_q and query mask M_q, we minimized the weighted logistic loss between the predicted mask $\hat{M}_q$ and M_q. We used stochastic gradient descent (SGD) with batch size being 1, learning rate 10^{-10} and momentum 0.99. We set the learning rate multiplier of the VGG branch (for generating classifier parameters) to 0.1 so that the two branches of our network converges in roughly the same speed. We trained the whole network in 60k iterations.

k-shot

With $k \neq 1$, the model jointly outputs k predicted masks for each image-mask pair in the support set. For simplicity and speed, each pixel in the final prediction mask is considered the foreground if it is classified as foreground in any of the k masks.

CHAPTER 4

DATASET, METRICS AND BASELINES

4.1 Dataset

To enlarge our dataset as much as possible, we created a new dataset, **PASCAL-5**, by combining PASCALVOC 2012 [21] and the portion of SDS that are not overlapping with PASCALVOC 2012 validation set [22]. We splited the 20 semantic classes into four groups $I = \{1, 2, 3, 4\}$, each of them contain 5 semantic classes, so that we could perform cross-validation of 4 folds. In the later sections, we will denote **PASCAL-5^i** as L_{test} being one of the semantic classes groups and L_{train} being the rest. The split of semantic classes is shown in 4.1.

$i = 0$	$i = 1$	$i = 2$	$i = 3$
aeroplane, bicycle, bird, boat, bottle	bus, car, cat, chair, cow	diningtable, dog, horse, motorbike, person	potted plant, sheep, sofa, train, tv/monitor

Table 4.1: PASCAL-5^i test classes. For the i^{th} fold, we excluded the corresponding testing classes and pick examples from PACAL training set on the remaining $15 + 1$ classes. For testing, we picked examples from test classes in the PASCAL validation. Thus, both classes and the data are different in training and testing.

4.2 Metric

We adapted the standard metric of meanIU for Image Segmentation problems to our **mean-IoU**. Specfically, we defined the per-class Intersection over Union (IoU_l) of semantic class l, given predicted binary masks $\{\hat{M}_q\}_{i=1}^{M}$ and ground-truth binary masks $\{M_q\}_{i=1}^{M}$, as:

$$IoU_l = \frac{tp_l}{tp_l + fp_l + fn_l}. \tag{4.1}$$

Here, tp_l is the number of true positives, fp_l is the number of false positives and fn_l is

the number of false negatives over the set of masks. **meanIoU** is therefore defined as

$$meanIoU = \frac{1}{N}\sum_{i=1}^{|L|} IoU_l \tag{4.2}$$

4.3 Baselines

Since one-shot image segmentation, we selected, modified or created several related baselines to evaluate our algorithms.

- **Base Classifiers** CNNs learn deep representations of images, so it is intuitive to apply classical machine learning algorithms using these features. Specifically, during training phase, we fine-tuned a FCN-32s pretrained on ILSVRC2014 data to perform 16-way (15 training foreground classes + 1 background class) pixel-wise predictions on the **PASCAL-5^i** dataset. During testing, we extracted feature vectors of each pixel in the support set, with which and their correspoding pixel-wise labels we trained some simple classifier. This classifier is later used to perform pixel-wise classification on feature vectors extracted from the query image. We experimented with 1-NN and logistic regression [1].

- **Fine-tuning** As suggested by [16], for each test iteration we fine-tune the trained FCN-16 on the support set and test on the query image. To avoid overfitting and to speed up inference, we only fine-tuned the fully connected layers (fc6, fc7, fc8). We also found that the fine-tuned network converges faster if we normalize the fc-7 features by a batch normalization layer.

- **Co-segmentation by Composition** A similar technique, unsupervised co-segmentation [23], utilizes low level information from images to segment out the common semantic class from set of images. We fed in $k+1$ images and obtain $k+1$ masks, but consider the last one as the predicted mask (with the last image as query image).

[1] We also trained linear SVM, but it is significantly worse than 1-NN

- **Siamese Network** Siamese Networks are shown to be useful in one-shot image classification task [24]. Inspired by their method [24], we used two branches of FCNs to extract dense features from the support and query images. In the test time, each pixel is labeled to its nearest neighbor in the support set, in the weighted L1 similarity metric. Cross-entropy is used as the loss function during training.

CHAPTER 5

EXPERIMENTS

5.1 Segmentation Performance

The performance of our proposed approach and baseline methods is shown in Table 5.1 and Table 5.2.

The results indicate that our approach can generalize on new classes better. And our approach is much better than other methods on one-shot learning task. Our method outperforms 1-NN and fine-tuning with $k = 1$ by 25% relative meanIoU. Qualitative results are shown in Figure 5.2.

As in the comparison with co-segmentation, Our method outperform the unsupervised co-segmentation by 16%, because we used strong pixel-level annotation. In fact, using only one strongly annotated can already beat the performance of image co-segmentation, shown in the last row of Table 5.2.

Dilated-FCN We also experimented with the dilated-FCN on **PASCAL-** $\mathbf{5^0}$ to examine the effect of resolution. The results are shown in Table 5.3. Though using dilated-FCN slightly improves the performance, it is costly to use dilated-FCN because of memory issue or limited computational power.

Running Time We recorded the running time of each algorithm in Table 5.4. We ran all experiments on a machine with a 4GHz Intel Core-i7 CPU, 32GB RAM, and a Titan X GPU. Our method is $\sim 3\times$ faster the than second fastest method logistic regression on one-shot segmentation task; on 5-shot segmentation task, it is $\sim 10\times$ faster.

Methods	PASCAL-5^0	PASCAL-5^1	PASCAL-5^2	PASCAL-5^3	Mean
1-NN	25.3	44.9	**41.7**	18.4	32.6
LogReg	26.9	42.9	37.1	18.4	31.4
Finetuning	24.9	38.8	36.5	30.1	32.6
Siamese	28.1	39.9	31.8	25.8	31.4
Ours	**33.6**	**55.3**	40.9	**33.5**	**40.8**

Table 5.1: One-shot semantic segmentation meanIoU on all folds. Please see Table 4.1 for the test class names.

Methods	PASCAL-5^0	PASCAL-5^1	PASCAL-5^2	PASCAL-5^3	Mean
Co-segmentation	25.1	28.9	27.7	26.3	27.1
1-NN	34.5	53.0	**46.9**	25.6	40.0
LogReg	**35.9**	51.6	44.5	25.6	39.3
Ours	**35.9**	**58.1**	42.7	**39.1**	**43.9**

Table 5.2: 5-shot semantic segmentation meanIoU on all folds. Please see Table 4.1 for the test class names.

Methods	1-shot	5-shot
Ours-32s	33.6	35.9
Ours-8s	37.0	37.43

Table 5.3: Dilated FCN meanIoU results on PASCAL-5^0 data.

Methods	1-shot	5-shot
1-NN	1.10	4.55
Logistic Reg	0.66	3.50
Finetune	5.56	-
Siamese	5.65	-
Ours-32s	0.19	0.21

Table 5.4: Inference Time (in sec).

5.2 Pretraining Effect

Even though we explicitly split all semantic classes into folds, the network can still learn some prior knowledge about these semantic classes during pre-training. In face, the semantic class labels in **PASCAL-5** are included in ILSVRC2014 dataset on which our network was pretrained on. Therefore, it is important to carefully examine the effects of these prior knowledge on the ability of generalization.

In this experiment, we used simpler AlexNet [2] architecture instead of VGG in both

branches of our network. We created two models using different pre-trained weights. For the first one we initialized the weights (AlexNet-771) with the pre-trained AlexNet model in [25], which is pretrained on a subset of ILSVRC2014 classes that does not have any intersection with PASCAL classes. This new dataset, which we denote as PASCAL-removed-ImageNet, consists of 771 categories in total. For the second one, AlexNet-1000, we intialize the weights with the Alexnet pre-trained on ImageNet.

In addition, we ran experiments on the logistic regression baseline. Similarly, here we ran logistic regression baselines on both Alexnet-1000 and Alexnet-771: we finetuned both networks on **PASCAL-5^0** to perform 16-way pixel-wise prediction. Then we trained classifiers on fc-7 features and use it to generate the predicted mask $\hat{M}_q$. Since we did not observed overfitting, we picked model at the 100k iteration to extract fc-7 features. We call these two baselines LogReg-1000 and LogReg-771. Because we used fixed networks (at 100k iteration), the performance of the baselines is presented as a horizontal line. The meanIoUs are plotted in Figure 5.1.

Our experiments support the results in [25] that shows similar generalization power in deep neural network for image classification task.

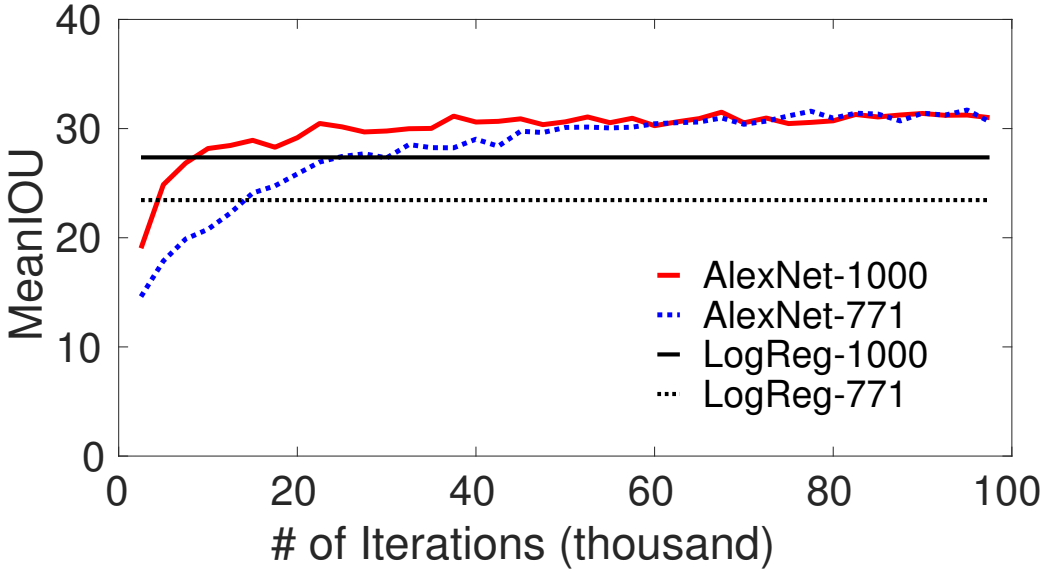

Figure 5.1: Comparison between AlexNet-1000 and AlexNet-771.

In Figure 5.1 we plotted the performance these networks. Initially, AlexNet-1000 shows better performance and converges faster. AlexNet-771 starts with lower meanIoU and converges slower. However, after convergence AlexNet-771 does almost the same as AlexNet-1000. The initial gap is mostly due to the fact that even classes in L_{train} were not presented during pre-training. AlexNet-771 can generalize to new classes that were not presented during pre-training and fine-tuning as well as AlexNet-1000 which has seen those classes during pre-training but not finetuning. Huh et al. [25] also reported similar results on object classification tasks. This result shows our network architecture can make predictions on brand-new categories as well as the categories observed during pretraining. In other words, the weak labels for brand new categories are not necessary for our proposed algorithm to obtain good performance. In contrast, LogReg-1000 outperforms LogReg-771, which implies that our logistic regression baseline cannot generalize well without those weak labels. This highlight the importance of meta-learning for one-shot image segmentation.

Figure 5.2: Qualitative results of our method in one-shot setting. Inside each tile, the image at the top is the support set and the large image is the query image. The ground-truth mask of the support set is overlaid with the annotation (in yellow). We overlaid the query image with the ground-truth mask in green and with the predicted mask in red. Overlap between prediction and ground truth appears yellow.

CHAPTER 6
CONCLUSION

In this thesis, we proposed a novel network architecture for one-shot image segmentation task. Our architecture successfully learns how to learn a classifier with which it perform pixel-wise prediction of each pixel in the query image. The experiments show the advantage of our design in both accuracy and speed.

REFERENCES

[1] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee. 2009, pp. 248–255.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[3] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[4] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[5] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3431–3440.

[7] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[8] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[9] Wei Liu et al. “SSD: Single shot multibox detector”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 21–37.

[10] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger”. In: *arXiv preprint arXiv:1612.08242* (2016).

[11] Fei-Fei Li, Rob Fergus, and Pietro Perona. “One-shot learning of object categories”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.4 (2006), pp. 594–611.

[12] Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. "One-shot learning with a hierarchical nonparametric Bayesian model". In: *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. 2012, pp. 195–206.

[13] Oriol Vinyals et al. "Matching networks for one shot learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3630–3638.

[14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[15] Anna Choromanska et al. "The Loss Surfaces of Multilayer Networks." In: *AISTATS*. 2015.

[16] Sergi Caelles et al. "One-Shot Video Object Segmentation". In: *Computer Vision and Pattern Recognition*. 2017.

[17] Anna Khoreva et al. *Lucid Data Dreaming for Object Tracking*. 2017. arXiv: 1703.09554.

[18] Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. "Image question answering using convolutional neural network with dynamic parameter prediction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 30–38.

[19] Wenlin Chen et al. "Compressing Neural Networks with the Hashing Trick." In: *ICML*. 2015, pp. 2285–2294.

[20] Fisher Yu and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[21] M. Everingham et al. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[22] Bharath Hariharan et al. "Simultaneous detection and segmentation". In: *European Conference on Computer Vision*. Springer. 2014, pp. 297–312.

[23] Alon Faktor and Michal Irani. "Co-segmentation by composition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1297–1304.

[24] Gregory Koch. "Siamese neural networks for one-shot image recognition". PhD thesis. University of Toronto, 2015.

[25] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. "What makes ImageNet good for transfer learning?" In: *arXiv preprint arXiv:1608.08614* (2016).